Data Quality: MiniProject

Khalid Belhajjame

Data Preparation Operations

Data Processing Operation	**Category**
Instance Selection	Horizontal Data Reduction
Drop Rows	
Undersampling	
Feature Selection	Vertical Data Reduction
Drop Columns	
Imputation	Data Transformation
Value transformation	
Binarization	
Normalization	
Discretization	
Instance Generation	Horizontal Data Augmentation
Oversampling	
Space Transformation	Vertical Data Augmentation
String Indexer	
One-Hot Encoder	
Join	Data Fusion
Append	

Objective

- Define a way to capture the provenance of the following operations using tensors in an efficient manner on large dataset
  - Fitler operation (horizontal reduction)
  - Oversampling (horizontal augmentation)
  - Join (data fusion)
  - Union (data fusion)
- Two methods for capturing the provenance:
  - Given an operation, the input dataset(s) and the output dataset, derive the tensor capturing the provenance
  - Modify the operation so as to make the capture of the provenancemore efficient
- We will be using sparse binary tensors

Data transformation

Data Transformation Operations in this category neither alter the schema of the dataset nor the number of records. Instead, they modify specific attribute values by applying a transformation function.

```
df['Age_Binarized'] = (df['Age'] >= 30).astype(int)
```

```
Original DataFrame:
      Name  Age  Salary
0    Alice   24   70000
1      Bob   17   40000
2  Charlie   35  120000
3    David   45  110000
4      Eve   19   50000

Transformed DataFrame with Binarized 'Age' column (1 if Age >= 30,
      Name  Age  Salary  Age_Binarized
0    Alice   24   70000              0
1      Bob   17   40000              0
2  Charlie   35  120000              1
3    David   45  110000              1
4      Eve   19   50000              0
```

Vertical Data Reduction

Vertical Reduction There are two operations that fall in this category, namely Feature Selection and Drop Columns. Both of this operations remove some of the attributes characterizing the data records in the input dataset D^{in} and produce a next dataset D^{out} that reflects the dataset obtained as a result.

```
df_reduced = df[['Name', 'Salary']]
```

```
Original DataFrame:
      Name  Age  Salary
0    Alice   24   70000
1      Bob   17   40000
2  Charlie   35  120000
3    David   45  110000
4      Eve   19   50000

Reduced DataFrame with selected features ('Name' and 'Salary'):
      Name  Salary
0    Alice   70000
1      Bob   40000
2  Charlie  120000
3    David  110000
4      Eve   50000
```

Horizontal Data Reduction

Horizontal Reduction Given a dataset D^{in}, an operation that performs horizontal reduction produces a new dataset D^{out}, where the data records in D^{out} are subsets of those in D^{in}: $D^{out} \subseteq D^{in}$. Data manipulations that fall into this category include the following operations: filtering, instance selection, row deletion, and undersampling.

```
df_reduced = df[df['Age'] >= 25]
```

```
Original DataFrame:
      Name  Age  Salary
0    Alice   24   70000
1      Bob   17   40000
2  Charlie   35  120000
3    David   45  110000
4      Eve   19   50000

Reduced DataFrame with rows where Age >= 25:
      Name  Age  Salary
2  Charlie   35  120000
3    David   45  110000
```

Vertical Data Augmentation

Vertical Data Augmentation Operations in this category include Space Transformation, String Indexer, and One-Hot Encoder. Given an input dataset D^{in}, applying vertical data augmentation produces a dataset D^{out} with a different schema from D^{in}. However, D^{in} and D^{out} have the same number of records, with the i^{th} record in D^{out} corresponding to the i^{th} record in D^{in}.

```
# Vertical data augmentation: Apply one-hot encoding to the Department column
df_augmented = pd.get_dummies(df, columns=['Department'])
```

```
Original DataFrame:
      Name   Department  Salary
0    Alice           HR   70000
1      Bob  Engineering  120000
2  Charlie  Engineering  115000
3    David    Marketing   90000
4      Eve           HR   65000

Augmented DataFrame with One-Hot Encoding:
      Name  Salary  Department_Engineering  Department_HR  Department_Marketing
0    Alice   70000                       0              1                     0
1      Bob  120000                       1              0                     0
2  Charlie  115000                       1              0                     0
3    David   90000                       0              0                     1
4      Eve   65000                       0              1                     0
```

Horizontal Data Augmentation

Horizontal Data Augmentation Operations that fall into this category are Instance Generation and Oversampling.

```
# Separate features (X) and target (y)
X = df[['Age', 'Salary']]  # Features
y = df['Category']         # Target (Class label)
```

```
# Horizontal data augmentation: Apply SMOTE for oversampling the minority class
smote = SMOTE(sampling_strategy='auto', random_state=42)
X_resampled, y_resampled = smote.fit_resample(X, y)
```

```
Original DataFrame:
      Name  Age  Salary Category
0    Alice   25   50000        A
1      Bob   30   60000        A
2  Charlie   35   70000        A
3    David   40   80000        B
4      Eve   45   90000        B
5    Frank   50  100000        B
6    Grace   55  110000        B
7   Hannah   60  120000        B

Augmented DataFrame after Oversampling (SMOTE):
   Age  Salary Category
0   25   50000        A
1   30   60000        A
2   35   70000        A
3   45   90000        B
4   50  100000        B
5   55  110000        B
6   60  120000        B
7   37   75000        A  # New synthetic sample generated by SMOTE
8   44   95000        B  # New synthetic sample generated by SMOTE
```

Join

Join The join of the datasets D^l and D^r, implemented using the Merge operation in the Pandas library, and denoted by $D^l \bowtie_C^t D^r$, produces a dataset D^j as a result of joining D^l and D^r on a boolean condition C, where t represents the join type (inner, left outer, right outer, or full outer).

Table 2.2: Dataset D^l

	ID	Birthdate	Gender	Postcode
1	10	1996-07-12	F	90210
2	20	1994-03-08	M	$\perp$
3	30	$\perp$	F	12345
4	40	1987-11-23	M	67890

Table 2.3: D^r Dataset

	ID	Name
1	20	Alice
2	40	Bob

Table 2.4: D^j dataset obtained by the following join $D^l \bowtie_{\text{inner}} D^r$

	ID	Birthdate	Gender	Postcode	Name
1	20	1994-03-08	M	$\perp$	Alice
2	40	1987-11-23	M	67890	Bob

Append

Append The append operation, implemented using *Concat* in the Pandas library, appends the records of a dataset D^l at the end of the D^r, denoted by $D^l \uplus D^r$. The two datsets do not need to have the same schema, and as such the results are extended with null for the mismatching attributes.

Table 2.2: Dataset D^l

	ID	Birthdate	Gender	Postcode
1	10	1996-07-12	F	90210
2	20	1994-03-08	M	$\perp$
3	30	$\perp$	F	12345
4	40	1987-11-23	M	67890

Table 2.3: D^r Dataset

	ID	Name
1	20	Alice
2	40	Bob

Table 2.5: D^a dataset obtained by the following append $D^l \uplus D^r$

	ID	Birthdate	Gender	Postcode	Name
1	10	1996-07-12	F	90210	$\perp$
2	20	1994-03-08	M	$\perp$	$\perp$
3	30	$\perp$	F	12345	$\perp$
4	40	1987-11-23	M	67890	$\perp$
5	20	$\perp$	$\perp$	$\perp$	Alice
6	40	$\perp$	$\perp$	$\perp$	Bob

Objective of the project

Overall goal: To develop a python class (which we could name tensprov) that can be used to infer the provenance of each of the operations just presented.

Given the input data frame(s), output data frame and the kind of the operation (vertical reduction, horizontal reduction, etc.), construct a tensor that informs on the provenance of the data records of the output data frames and how they depends on the input data frames.

We will be using **binary sparse tensors**.

We will be using tensors, specifically binary sparse tensors, to capture the provenance

Join The join of the datasets D^l and D^r, implemented using the Merge operation in the Pandas library, and denoted by $D^l \bowtie_C^t D^r$, produces a dataset D^j as a result of joining D^l and D^r on a boolean condition C, where t represents the join type (inner, left outer, right outer, or full outer).

Table 2.2: Dataset D^l

	ID	Birthdate	Gender	Postcode
1	10	1996-07-12	F	90210
2	20	1994-03-08	M	$\perp$
3	30	$\perp$	F	12345
4	40	1987-11-23	M	67890

Table 2.3: D^r Dataset

	ID	Name
1	20	Alice
2	40	Bob

$$T = \left(\begin{pmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 0 \\ 0 & 0 \end{pmatrix}, \begin{pmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 1 \end{pmatrix} \right)$$

Table 2.4: D^j dataset obtained by the following join $D^l \bowtie_{\text{inner}} D^r$

	ID	Birthdate	Gender	Postcode	Name
1	20	1994-03-08	M	$\perp$	Alice
2	40	1987-11-23	M	67890	Bob

Usage of the tensprov class

 # Create TensProv object

```
tensprov = TensProv()
```

 # Perform the join

```
Do = Dl.merge(Dr, how='inner', left_on='key', right_on='key', suffixes=('_left', '_right’))
```

 # Capture provenance for the join operation

```
tensprov.construct(Dl, Dr, Do, operation_type='join')
```

Tasks

- Develop the tensprov class
- Assess its performance in term of processing time
- Many alternatives are possible for deriving the provenance, we will discuss some of them.
  - You should imlplement at least two alternatives for each typoe of data processing operation
- I will send you (large) datasets that can be used to evaluate the performance.

Inferring provenance by examining the data

- For certain analysis, like data transformation, vertical reduction and vertical augmentation, we do not need to examine the data, we can directly generate a diagonal bi-binary tensor where the two dimensions are equal to the size of the dataset subject to manipulation

```
df['Age_Binarized'] = (df['Age'] >= 30).astype(int)
```

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

```
Original DataFrame:
      Name  Age  Salary
0    Alice   24   70000
1      Bob   17   40000
2  Charlie   35  120000
3    David   45  110000
4      Eve   19   50000

Transformed DataFrame with Binarized 'Age' column (1 if Age >= 30,
      Name  Age  Salary  Age_Binarized
0    Alice   24   70000              0
1      Bob   17   40000              0
2  Charlie   35  120000              1
3    David   45  110000              1
4      Eve   19   50000              0
```

Inferring provenance by examining the data

- For the remaining types of data processing operations, i.e., horiziontal data reduction, horizontal data augmentation, join and append, we do need to examine the data to generate the tensors capturing the provenance

```python
df_reduced = df[df['Age'] >= 25]
```

```
Original DataFrame:
      Name  Age  Salary
0    Alice   24   70000
1      Bob   17   40000
2  Charlie   35  120000
3    David   45  110000
4      Eve   19   50000

Reduced DataFrame with rows where Age >= 25:
      Name  Age  Salary
2  Charlie   35  120000
3    David   45  110000
```

Deriving provenance using hashing

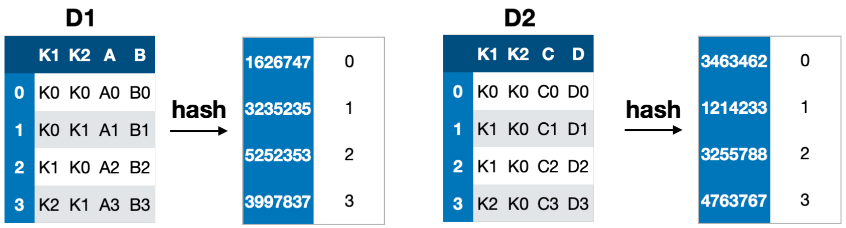
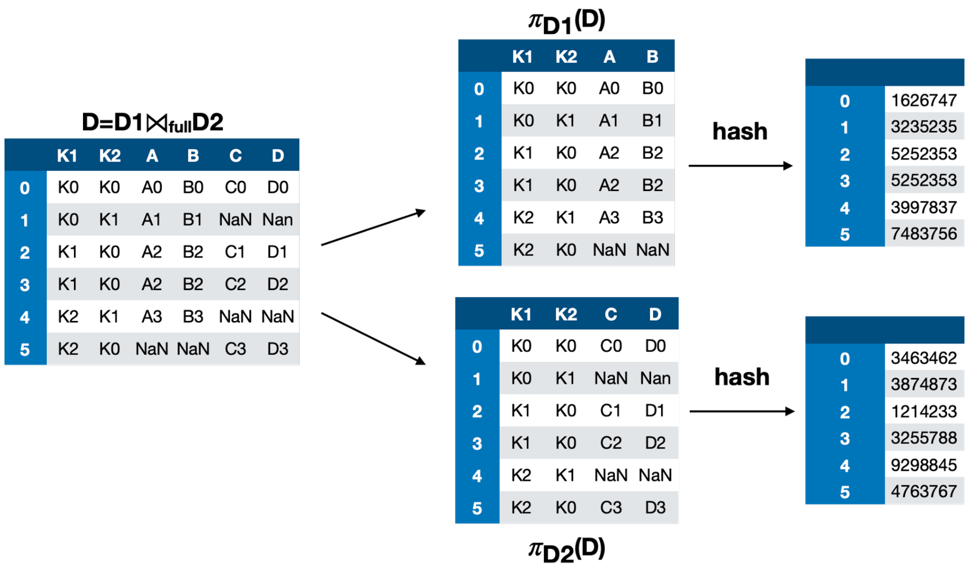


Figure 15: Hashing of the input dataframes



Recomputing the join using the key attributes

Hashing techniques can be computationally costly, especially when the input or output datasets are large. A more efficient approach for reconstructing the provenance of a join operation involves the following steps: First, add an ID column to each input dataset to uniquely identify their data records. Th IDs coresponds to the index of the data recorss. Also, the attributes that do not participate in the join (except the ID columns) are projected out. Next, the join operation as originally intended is performed between the obtained datasets. The resulting output dataset will contain, in addition to the join attributes, the IDs of the records from each input dataset that were joined. These IDs can then be used directly to trace the provenance of each output record.

Other methods for reconstructing the provenance ?

Organization

- You will work in teams of 4 or 5 teams
- You are expected to return a projevct code and a report describing what you did. The quality of the code matters.
- You will need to test your code and report on the performance of your method
- In the last session of the course, each team will be presenting their solution.